

Dokumentacja techniczna programu „Biblioteka”



Stworzyli:
Patryk Szalaśny
Bartosz Wężyk

Spis treści

Wstęp	2
Podstawowe założenia	2
Wstępnie o budowie programu	2
MySQL „Biblioteka”	2
Zależności	2
Działanie programu/formularze	3
Główny.java	3
PolaczenieBaza.java	3
Logowanie.java	3
Stronadomowa.java	4
Przyciski funkcji	4
Dodaj użytkownika/dodaj książkę	4
Edycja użytkowników/Edycja książek	5
Listing danych	6
Usuwanie użytkowników/usuwanie książek	6
Blokowanie użytkowników	7
Wypożyczanie/zwrot książek	7
Listing przeterminowanych wypożyczeń	8

Wstęp

Podstawowe założenia

Stworzona przez nas aplikacja „Biblioteka” jest prostym programem służącym do zarządzania zasobami biblioteki. Program został napisany w języku Java przy użyciu wbudowanych do programu ramek (frames). Do działania program wykorzystuje bazę danych MySQL o takiej samej nazwie tj. „Biblioteka”.

Wstępnie o budowie programu

Cały program oparty jest na 14 pomniejszych komponentach, które zostały stworzone w celu wykonywania pożądaných przez użytkownika funkcji m.in. dodawania i usuwania użytkowników oraz książek czy wypożyczania książek z biblioteki. Program oparty jest na formularzu „Logowanie” z którego następnie przenosimy się do strony głównej, na której znajdują się poszczególne komponenty programu.

MySQL „Biblioteka”

Do działania programu wykorzystywana jest baza MySQL, w której znajdują się 3 tabele: użytkownik, książka oraz wypożyczenia.

```
+-----+
| Tables_in_biblioteka |
+-----+
| książka               |
| użytkownik            |
| wypożyczenia          |
+-----+
3 rows in set (0.00 sec)
```

Każda z nich dzieli się na kilka kolumn:

```
mysql> desc książka;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| tytuł          | varchar(30)   | YES  |     | NULL    |       |
| autor          | varchar(30)   | YES  |     | NULL    |       |
| wydawnictwo    | varchar(20)   | YES  |     | NULL    |       |
| numerkatalogowy | varchar(10)   | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+

mysql> desc użytkownik;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| imię           | varchar(10)   | YES  |     | NULL    |       |
| nazwisko       | varchar(15)   | YES  |     | NULL    |       |
| numerkarty     | varchar(10)   | NO   | PRI | NULL    |       |
| numertelefonu  | varchar(9)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+

mysql> desc wypożyczenia;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| numerkatalogowy | varchar(10)   | NO   | PRI | NULL    |       |
| numerkarty      | varchar(10)   | YES  |     | NULL    |       |
| datawydania     | date          | YES  |     | NULL    |       |
| datazwrotu      | date          | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

Zależności

Program zaczyna od okna logowania, w którym prosi na o podanie nazwy użytkownika oraz hasła. Po udanym zalogowaniu przenosi nas do strony domowej. Z formularzu „Stronadomowa” mamy dostęp do wszystkich niżej wymienionych funkcji programu. Formularz składa się z 13 przycisków, które po naciśnięciu otwierają okno z wybraną funkcją opisaną na przycisku.

Działanie programu/formularze

Główny.java

Prosty plik zawierający w sobie klasę **main** służy do uruchamiania formularzu „Logowanie”. Został napisany w celu wyeliminowania możliwości pojawiania się problemu z uruchamianiem pliku Biblioteka.exe.

```
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.UIManager; import
javax.swing.UnsupportedLookAndFeelException; import
javax.swing.plaf.nimbus.NimbusLookAndFeel;

class Główny {
    public static void main(String[] args) {
        Logowanie logowanie = new Logowanie();
        try {
            UIManager.setLookAndFeel(new NimbusLookAndFeel());
        } catch (UnsupportedLookAndFeelException ex) {
            Logger.getLogger(Logowanie.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

PołączenieBaza.java

Ważną częścią programu jest plik „PołączenieBaza.java”, jest to krótki plik pozwalający na połączenie się poszczególnych komponentów programu z bazą danych MySQL. Jego główną funkcją jest przyspieszenie i ułatwienie pracy.

```
package Projekt;
import java.sql.*;

public class PołączenieBaza {
    public static Connection getCon() {
        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/Biblioteka?useUnicode=true&characterEncoding=UTF-8","root","admin");
            return con;
        }catch(Exception e){
            System.out.println(e);
            return null;
        }
    }
}
```

Te kilka linijek pozwala nam na łączenie się z bazą danych przez zaimportowanie pliku „PołączenieBaza.java” przy użyciu **import Projekt.PołączenieBaza;** do każdego z formularzy programu, co znacznie skraca kod i w wyniku zwiększa jego czytelność.

Logowanie.java

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(jTextField1.getText().equals("admin") && jPasswordField1.getText().equals("admin"))
    {
        setVisible(false);
        new Stronadomowa().setVisible(true);
    }
    else
        JOptionPane.showMessageDialog(null, "Niepoprawne hasło lub nazwa użytkownika");
    jTextField1.setText("");
    jPasswordField1.setText("");
}
```

Oto prosty kod dla przycisku Zaloguj, który sprawdza czy w polach tekstowych znajdują się właściwe dane wejściowe.

W naszym przypadku jest to nazwa użytkownika „**admin**” oraz hasło „**admin**”. Jeśli podana nazwa użytkownika oraz hasło są prawidłowe program zamyka okno logowania i przenosi nas do strony domowej. W przypadku podania błędnej nazwy użytkownika lub hasła program wyświetla na ekranie komunikat „Niepoprawne hasło lub nazwa użytkownika”, czyści pola i pozwala na podanie danych jeszcze raz.

Stronadomowa.java

Formularz strony domowej zawiera w sobie przyciski, gdzie każdy z nich ma przypisaną do siebie prostą komendę `newNAZWAFORMULARZA().setVisible(true)`, która służy do otwierania kolejnych formularzy w zależności od potrzeb użytkownika oraz przycisk wyloguj, który cofa nas do formularza „Logowanie”.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    setVisible(false);  
    new Logowanie().setVisible(true);  
}  
  
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Nowyuzownik().setVisible(true);  
}  
  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Nowaksiazka().setVisible(true);  
}  
  
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Usunuzownika().setVisible(true);  
}  
  
private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Usunksiazke().setVisible(true);  
}  
  
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Wypozyczenie().setVisible(true);  
}  
  
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Zwrot().setVisible(true);  
}  
  
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {  
    new ListWypozyczone().setVisible(true);  
}  
  
private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {  
    new ListKsiazki().setVisible(true);  
}  
  
private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {  
    new ListUzytkownicy().setVisible(true);  
}  
  
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Edytujuzownika().setVisible(true);  
}  
  
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {  
    new Edytujksiazke().setVisible(true);  
}
```

Przyciski funkcji

Dodaj użytkownika/dodaj książkę

Zarówno formularz dodawania użytkownika oraz dodawania książki działają na bazie tego samego kodu.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String tytuł=jTextField1.getText();  
    String autor=jTextField2.getText();  
    String wydawnictwo=jTextField3.getText();  
    String numerkatalogowy=jTextField4.getText();  
    try  
    {  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        st.executeUpdate("insert into ksiazka values('"+tytuł+"','"+autor+"','"+wydawnictwo+"','"+numerkatalogowy+"')");  
        JOptionPane.showMessageDialog(null,"Dodano pozycję");  
        setVisible(false);  
        new Nowaksiazka().setVisible(true);  
    }  
    catch(Exception e)  
    {  
        JOptionPane.showMessageDialog(null,"Książka o podanym numerze katalogowym już istnieje w bazie");  
        setVisible(false);  
        new Nowaksiazka().setVisible(true);  
    }  
}
```

Gdzie pierwsze 4 linijki odpowiadają za przypisanie poszczególnym polom tekstowym nazw tj. dla dodawania książek „tytuł”, „autor”, „wydawnictwo” oraz „numerkatalogowy”. Następnie używając bloku try-catch kolejno program łączy się z bazą używając wcześniej stworzonego pliku „PolaczenieBaza.java”, po czym dzięki klasie Statement jest w stanie wysłać zapytanie do bazy MySQL. Używając polecenia „st.executeUpdate(„...”)” wysyła do bazy polecenia zaktualizowania tabeli „ksiazka” używając po kolei wcześniej zdefiniowanych zmiennych String. Na koniec wyświetla komunikat o pomyślnym dodaniu książki do bazy danych, zamyka okno dodawania i tworzy nowe, „czyste”, które jest gotowe do kolejnego dodania. W przypadku, gdy program nie jest w stanie wykonać naszego zapytania wyświetla komunikat o błędzie.

Formularz dodawania użytkowników działa na tej samej zasadzie, jedynymi różnicami są nazwy zmiennych jakie wprowadzamy do formularza oraz tabela na której operuje formularz.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String imię=jTextField1.getText();  
    String nazwisko=jTextField2.getText();  
    String numerkarty=jTextField3.getText();  
    String numertelefonu=jTextField4.getText();  
    try  
    {  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        st.executeUpdate("insert into uzytkownik values('"+imię+"','"+nazwisko+"','"+numerkarty+"','"+numertelefonu+"')");  
        JOptionPane.showMessageDialog(null,"Dodano użytkownika");  
        setVisible(false);  
        new Nowyuzzytkownik().setVisible(true);  
    }  
    catch(Exception e)  
    {  
        JOptionPane.showMessageDialog(null,"Numer karty użytkownika już istnieje w systemie");  
        setVisible(false);  
        new Nowyuzzytkownik().setVisible(true);  
    }  
}
```

Edycja użytkowników/Edycja książek

Tak jak w przypadku poprzednich formularzy edycja użytkowników i książek jest oparta na takim samym kodzie.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String imię=jTextField1.getText();  
    String nazwisko=jTextField2.getText();  
    String numerkarty=jTextField3.getText();  
    String numertelefonu=jTextField4.getText();  
  
    try{  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        st.executeUpdate("update uzytkownik set imię='"+imię+"',nazwisko='"+nazwisko+"',numertelefonu='"+numertelefonu+"' where numerkarty='"+numerkarty+"'");  
        JOptionPane.showMessageDialog(null, "Pomyślnie zaktualizowano rekord bazy");  
        setVisible(false);  
        new Edytujuzzytkownika().setVisible(true);  
    }catch(Exception e){  
        JOptionPane.showMessageDialog(null, "Nie można zaktualizować danych");  
        setVisible(false);  
        new Edytujuzzytkownika().setVisible(true);  
    }  
}
```

Pierwsze 4 linijki odpowiadają za przypisanie poszczególnym polom tekstowym nazw tj. dla edycji użytkowników „imię”, „nazwisko”, „numerkarty” oraz „numertelefonu”. Następnie używając bloku try-catch kolejno program łączy się z bazą używając wcześniej stworzonego pliku „PolaczenieBaza.java”, po czym dzięki klasie Statement jest w stanie wysłać żądania do bazy MySQL. Używając polecenia „st.execute („...”)” wysła do bazy polecenia zaktualizowania tabeli „uzytkownik” używając po kolei wcześniej zdefiniowanych zmiennych String. Jednak tym razem w nawiasie dodane jest polecenie „update”, które zamiast dodawać użytkowników do tabeli, aktualizuje już istniejące rekordy tabeli używając wybranego parametru do rozpoznania, który z rekordów chcemy zaktualizować. W naszym przypadku jest to pole „numerkarty”, jako że zostało ono przez nas ustawione z własnością **primary key** i jest niepowtarzalne dla każdego z użytkowników biblioteki. Na koniec wyświetla komunikat o pomyślnym dodaniu książki do bazy danych, zamyka okno dodawania i tworzy nowe, „czyste”, które jest gotowe do kolejnego dodania. W przypadku, gdy program nie jest w stanie wykonać naszego żądania wyświetla komunikat o błędzie.

Formularz edycji książek działa na tej samej zasadzie, jedynymi różnicami są nazwy zmiennych jakie wprowadzamy do formularza oraz tabela na której operuje formularz.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String tytul=jTextField1.getText();  
    String autor=jTextField2.getText();  
    String wydawnictwo=jTextField3.getText();  
    String numerkatalogowy=jTextField4.getText();  
  
    try{  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        st.executeUpdate("update ksiazka set tytul='"+tytul+"',autor='"+autor+"',wydawnictwo='"+wydawnictwo+"' where numerkatalogowy='"+numerkatalogowy+"'");  
        JOptionPane.showMessageDialog(null, "Pomyślnie zaktualizowano rekord bazy");  
        setVisible(false);  
        new Edytujuzzytkownika().setVisible(true);  
    }catch(Exception e){  
        JOptionPane.showMessageDialog(null, "Nie można zaktualizować danych");  
        setVisible(false);  
        new Edytujuzzytkownika().setVisible(true);  
    }  
}
```

Listing danych

Wszystkie formularze listingu danych oparte są o ten sam kod z dodanymi małymi modyfikacjami w zależności od funkcji jaką mają spełniać. W celu stworzenia formularza wyświetlającego dane posortowane w zależności od wybranego przez nas parametru, program wykorzystuje instrukcję warunkową switch (screen poniżej).

```
public ListUzytkownicy(String tytul, int i) {
    initComponents();

    try {
        Connection con=PolaczenieBaza.getCon();
        Statement st=con.createStatement();
        String parametr = "";
        switch (i) {
            case 1:
                parametr = "imię";
                break;
            case 2:
                parametr = "nazwisko";
                break;
            case 3:
                parametr = "numerkarty";
                break;
            case 4:
                parametr = "numertelefonu";
                break;
            default:
                break;
        }

        String sql = "select * from uzytkownik where "+parametr+"='"+tytul+"'";
        ResultSet rs = st.executeQuery(sql);

        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        model.setRowCount(0);

        while(rs.next()){
            String tytul = rs.getString("imię");
            String autor = rs.getString("nazwisko");
            String wydawnictwo = rs.getString("numerkarty");
            String numerkatalogowy = rs.getString("numertelefonu");
            String zablokowany = rs.getString("zablokowany");

            String tbData[]={tytul,autor,wydawnictwo,numerkatalogowy,zablokowany};
            DefaultTableModel tblModel = (DefaultTableModel) jTable1.getModel();

            tblModel.addRow(tbData);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Wysąpił błąd");
    }
}
```

W zależności od naciśniętego przez nas przycisku program sortuje dane zależnie od imienia, nazwiska, numeru karty lub numer telefonu użytkownika.

Usuwanie użytkowników/usuwanie książek

Tak jak w przypadku poprzednich formularzy formularz usuwania jest oparty na przycisku, który zawiera następujący kod

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    String numerkarty=jTextField1.getText();
    try{
        Connection con=PolaczenieBaza.getCon();
        Statement st=con.createStatement();
        st.executeUpdate("DELETE FROM uzytkownik WHERE numerkarty = " + numerkarty);
        JOptionPane.showMessageDialog(null,"Usunięto rekord z bazy danych");
        setVisible(false);
        new Usun uzytkownika().setVisible(true);
        con.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,"Podany numer nie istnieje");
        setVisible(false);
        new Usun uzytkownika().setVisible(true);
    }
}
```

Jest to bardzo prosty formularz, w którym jedyną zmienną podawaną przez osobę zarządzającą jest numer karty użytkownika. Następnie używając bloku try-catch program łączy się z bazą, tworzy klasę Statement w celu wysyłania żądań i dzięki poleceniu „st.executeUpdate(„DELETE FROM uzytkownik WHERE numerkarty = ” + numerkarty);” usuwana użytkowników w zależności od podanego numerkarty. Na koniec wyświetla komunikat o pomyślnym usunięciu rekordu bazy, zamyka okno usuwania i tworzy nowe, „czyste” okno, gotowe do pracy. W wypadku wystąpienia błędu program wyświetla komunikat o braku możliwości usunięcia rekordu bazy.

Formularz usuwania książek działa na bazie tego samego kodu. Zmieniona została jedynie nazwa zmiennej oraz tabela z której usuwane są dane.

Blokowanie użytkowników

Formularz blokowanie użytkowników jest oparty o prosty kod, który po połączeniu się z bazą wybiera użytkownika według wpisanego przez nas numeru karty, a następnie ustawia wartość zmiennej „zablokowany” na 1 co oznacza zablokowanie użytkownikowi możliwości wypożyczania książek z biblioteki.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String c = jTextField1.getText();  
  
    try  
    {  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        ResultSet d = st.executeQuery("SELECT * FROM uzytkownik WHERE numerkarty='"+c+"'");  
        if (d.next() == true) {  
            st.executeUpdate("UPDATE uzytkownik SET zablokowany=1 WHERE numerkarty='"+c+"'");  
            JOptionPane.showMessageDialog(null,"Zablokowano użytkownika");  
        } else if (d.next() == false) {  
            JOptionPane.showMessageDialog(null,"Taki użytkownik nie istnieje");  
        }  
    }  
    catch(Exception e)  
    {  
        JOptionPane.showMessageDialog(null,"Nie udało się zablokować");  
    }  
}
```

Wypożyczanie/zwrot książek

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String numerkatalogowy=jTextField1.getText();  
    String numerkarty=jTextField2.getText();  
    String datawydania=jTextField3.getText();  
    String datazwrotu=jTextField4.getText();  
    try{  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        ResultSet rs=st.executeQuery("select * from ksiazka where numerkatalogowy='"+numerkatalogowy+"'");  
        if(rs.next())  
        {  
            ResultSet rs1=st.executeQuery("select * from uzytkownik where numerkarty='"+numerkarty+"'");  
            if (rs1.next()) {  
                int w = Integer.parseInt(rs1.getString(5));  
                if (w!=1) {  
                    st.executeUpdate("insert into wypozyczenia values('"+numerkatalogowy+"','"+numerkarty+"','"+datawydania+"','"+datazwrotu+"')");  
                    JOptionPane.showMessageDialog(null,"Wydano książkę");  
                    setVisible(false);  
                    new Wypozyczenie().setVisible(true);  
                } else  
                {  
                    JOptionPane.showMessageDialog(null, "Ten użytkownik jest zablokowany");  
                }  
            }  
            else  
            {  
                JOptionPane.showMessageDialog(null, "Nieścisły numer karty użytkownika");  
            }  
        }  
        else  
        {  
            JOptionPane.showMessageDialog(null, "Nieścisły numer katalogowy książki");  
        }  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Nie można wydać książki");  
        setVisible(false);  
        new Wypozyczenie().setVisible(true);  
    }  
}
```

Za wypożyczanie książek odpowiada przycisk w formularzu, którego kod wykonuje następujące czynności.

Pierwsze 4 linijki odpowiadają za przypisanie poszczególnym polom tekstowym nazw tj. dla edycji użytkowników „imię”, „nazwisko”, „numerkarty” oraz „numertelefonu”. Następnie używając bloku try-catch kolejno program łączy się z bazą używając wcześniej stworzonego pliku „PolaczenieBaza.java”, po czym dzięki klasie Statement jest w stanie wysłać żądania do bazy MySQL. Używając polecenia „st.executeQuery („...”)” wysła do bazy „zapytanie” o istnienie numeru katalogowego danej książki, następnie sprawdza czy w bazie istnieje podany przez nas numer karty użytkownika, a na koniec wykorzystując polecenie „st.executeUpdate(„INSERT INTO ...);” dodaje do tabeli wypozyczenia, znajdującej się w bazie biblioteka, rekord, w którym zapisane są numer katalogowy wypożyczonej książki, numer karty użytkownika, który wypożyczył daną pozycję, data wypożyczenia oraz termin zwrotu danej książki.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String numerkatalogowy=jTextField1.getText();  
    String numerkarty=jTextField2.getText();  
    try{  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        st.executeUpdate("DELETE FROM wypozyczenia WHERE numerkatalogowy = " + numerkatalogowy);  
        JOptionPane.showMessageDialog(null,"Usunięto rekord z bazy danych");  
        setVisible(false);  
        new Zwrot().setVisible(true);  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Podany numer nie istnieje");  
        setVisible(false);  
        new Zwrot().setVisible(true);  
    }  
}
```

Zwrot książek działa w bardzo prosty sposób. Jest to po prostu formularz usuwający rekordy z tabeli wypozyczenia.

Listing przeterminowanych wypożyczeń

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    try {  
        Connection con=PolaczenieBaza.getCon();  
        Statement st=con.createStatement();  
        String sql = "SELECT * FROM wypozyczenia WHERE datazwrotu < CURDATE()";  
        ResultSet rs = st.executeQuery(sql);  
  
        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();  
        model.setRowCount(0);  
  
        while(rs.next()){  
            String numerkatalogowy = rs.getString("numerkatalogowy");  
            String numerkarty = rs.getString("numerkarty");  
            String datawydania = rs.getString("datawydania");  
            String datazwrotu = rs.getString("datazwrotu");  
  
            String tbData[]={numerkatalogowy,numerkarty,datawydania,datazwrotu};  
            DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();  
  
            tblModel.addRow(tbData);  
        }  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null, "Wysąpił błąd");  
    }  
}
```

Listing wyświetlający książki wypożyczone, które nie zostały zwrócone w terminie. Jego działanie polega na tym, że po połączeniu z bazą danych, program porównuje kolumnę „datazwrotu” tabeli wypożyczone z datą systemową. W przypadku, gdy data terminu zwrotu jest starsza od daty systemowej program wyświetli książki, które nie zostały zwrócone do biblioteki w terminie.

